

Apple

\$1.80



Assembly

Line

Volume 6 -- Issue 4

January, 1986

Convert Lo-Res Pictures to Hi-Res.	2
A Question About BRUN.	10
Monthly AAL Source Disk Subscriptions.	11
Text File Transfer Using DOS 3.3 File Manager.	12
Fast 6502 & 65802 Multiply Routines.	18
RAMWORKS Compatible Auxiliary MOVE Routine	27
Correction to DOS/ProDOS Double INIT	32
An Interesting Bit of Trivia	32

New Low Diskette Price

When we first started offering blank diskettes for sale, back in May of 1981, we were able to sell them for the then below retail price of only \$50 for 20 diskettes. There have been quite a few changes in this industry in the last five years, and prices have continued to fall. We have a new supplier and can now offer you quality diskettes for only \$20 per package of 20, a reduction of 37% since last month.

New Monthly Disks

Since diskette prices have fallen so far, we are now planning to send out disks containing the source code from AAL on a monthly basis, instead of quarterly as we have in the past. See the note on page 11 for the details.

Discover

Sears Financial Network is launching a new credit card this year, called Discover. They offer lower interest rates and no yearly charge to the consumer, and better rates to the merchants as well, so we are pleased to be able to accept this card now. You can use it just like any other card for your phone and mail orders.

Convert Lo-Res Pictures to Hi-res....Bob Sander-Cederlof

Most Apple dot-matrix printer interfaces now include the firmware to print hi-res graphics pictures directly from a screen image. However, most do not provide any way to print lo-res graphics pictures. With the program presented here you can convert a lo-res graphics image into a hi-res picture, ready to be printed by your interface firmware.

Even if you don't ever plan to do such a thing, there are some neat coding tricks in the following program, which you might be able to apply to other hi-res programs.

Lines 1070-1200 demonstrate the use of my lo- to hi-res conversion program. Notice that I started with the label "T". I find I am using that label all the time, lately. I think I started using it as a short mnemonic for "TEST". It is convenient, because in the S-C Macro Assembler environment I can test the program I just assembled by typing "MGO" and the label I want to start at. I find my fingers can now type "MGOT" without my brain even realizing it happened.

The first thing my demo does is call PLOT to create a lo-res picture. I didn't have any real lo-res art around, so I simply drew a 4-by-4 pattern showing all 16 lo-res colors. PLOT fills 16 (4x4) pixels with color 0, 16 with color 1, and so on:

lo-res	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	4	4	4	4	8	8	8	8	C	C	C	C
1	0	0	0	0	4	4	4	4	8	8	8	8	C	C	C	C
2	0	0	0	0	4	4	4	4	8	8	8	8	C	C	C	C
3	0	0	0	0	4	4	4	4	8	8	8	8	C	C	C	C

15	3	3	3	3	7	7	7	7	B	B	B	B	F	F	F	F

The rest of the lo-res screen I did not change, so it will normally show the lo-res equivalent of whatever text was on the screen. Of course if you were really trying to use my CONVERT program you would draw your real lo-res picture.

Lines 1090-1120 turn on the lo-res graphics display, and line 1130 waits until I press a key on the keyboard. After running this much of the program, and studying the dot patterns on the screen, I realized that it is not possible to exactly reproduce the lo-res colors on the hi-res screen (unless I used //e or //c double hi-res). However, by mixing various patterns of dots within the 28 dots (7x4) each lo-res pixel maps to, I can come close to the same color. I don't really know how close I came, because I do not have a color monitor. However, I can at least tell by inspection that all 16 colors map to different dot patterns that will be distinguishable colors.

The PAUSE.FOR.ANY.KEY subroutine will return EQ status if the key I press is RETURN, and NE status if it is any other key. Line 1140 will terminate my test program if RETURN was typed. If it was not RETURN, line 1150 turns on hi-res graphics and line 1160 calls the convert program. Then line 1170 waits for

S-C Macro Assembler Version 2.0.....DOS \$100, ProDOS \$100, both for \$120
 ProDOS Upgrade Kit for Version 2.0 DOS owners.....\$30
 Version 2.0 Upgrade Kit for 1.0/1.1/1.2 owners.....\$20
 Source Code of S-C Macro 2.0 (DOS only).....additional \$100
 Full Screen Editor for S-C Macro (with complete source code).....\$49
 S-C Cross Reference Utility.....without source code \$20, with source \$50
 RAK-Ware DISASM.....\$30
 Source Code for DISASM.....additional \$30
 S-C Word Processor (with complete source code).....\$50
 DPL8 Source and Object.....\$50
 Double Precision Floating Point for Applesoft (with source code).....\$50
 "Bag of Tricks", Worth & Lechner, with diskette.....(\$39.95) \$36 *
 MacASM -- Macro Assembler for Macintosh (Mainstay).....(\$150.00) \$100 *
 S-C Documentor (complete commented source code of Applesoft ROMs).....\$50
 Source Code of //e CX & F8 ROMs on disk.....\$15
 Cross Assemblers for owners of S-C Macro Assembler.....\$32.50 to \$50 each
 (Available: 6800/1/2, 6301, 6805, 6809, 68000, Z-80, Z-8, 8048,
 8051, 8085, 1802/4/5, PDP-11, GIL650/70, others)

AAL Quarterly Disks.....each \$15, or any four for \$45
 Each disk contains the source code from three issues of AAL,
 saving you lots of typing and testing.
 The quarters are Jan-Mar, Apr-Jun, Jul-Sep, and Oct-Dec.
 (All source code is formatted for S-C Macro Assembler. Other assemblers
 require some effort to convert file type and edit directives.)

Diskettes (with hub rings)..... package of 20 for \$20 *
 Vinyl disk pages, 6"x8.5", hold two disks each.....10 for \$6 *
 Diskette Mailing Protectors (hold 1 or 2 disks).....40 cents each
 (Cardboard folders designed to fit 6"x9" Envelopes.) or \$25 per 100 *
 Envelopes for Diskette Mailers..... 6 cents each

65802 Microprocessor (Western Design Center).....(\$95) \$50 *
 quikLoader EPROM System (SCRG).....(\$179) \$170 *
 PROMGRAMMER (SCRG).....(\$149.50) \$140 *
 Switch-a-Slot (SCRG).....(\$179.50) \$170 *
 Extend-a-Slot (SCRG).....(\$35) \$32 *

"Programming the 65816", Eyes.....(\$22.95) \$21 *
 "Apple //e Reference Manual", Apple Computer.....(\$24.95) \$23 *
 "Apple //c Reference Manual", Apple Computer.....(\$24.95) \$23 *
 "ProDOS Technical Reference Manual", Apple Computer.....(\$29.95) \$27 *
 "Now That You Know Apple Assembly Language...", Gilder.....(\$19.95) \$18 *
 "Apple ProDOS: Advanced Features for Programmers", Little..(\$17.95) \$17 *
 "Inside the Apple //c", Little.....(\$19.95) \$18 *
 "Inside the Apple //e", Little.....(\$19.95) \$18 *
 "Apple II+/IIf Troubleshooting & Repair Guide", Brenner.....(\$19.95) \$18 *
 "Apple II Circuit Description", Gayler.....(\$22.95) \$21 *
 "Understanding the Apple II", Sather.....(\$22.95) \$21 *
 "Understanding the Apple //e", Sather.....(\$24.95) \$23 *
 "Enhancing Your Apple II, vol. 1", Lancaster.....(\$15.95) \$15 *
 "Enhancing Your Apple II, vol. 2", Lancaster.....(\$17.95) \$17 *
 "Assembly Cookbook for the Apple II/IIf", Lancaster.....(\$21.95) \$20 *
 "Beneath Apple DOS", Worth & Lechner.....(\$19.95) \$18 *
 "Beneath Apple ProDOS", Worth & Lechner.....(\$19.95) \$18 *
 "Real Time Programming -- Neglected Topics", Foster.....(\$9.95) \$9 *
 "Microcomputer Graphics", Myers.....(\$12.95) \$12 *
 "Assem. Language for Applesoft Programmers", Finley & Myers.(\$16.95) \$16 *
 "Assembly Lines -- the Book", Wagner.....(\$19.95) \$18 *
 "AppleVisions", Bishop & Grossberger.....(\$39.95) \$36 *

* On these items add \$2.00 for the first item and
 \$.75 for each additional item for US shipping.
 Foreign customers inquire for postage needed.

Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
 *** (214) 324-2050 ***
 *** We accept Master Card, VISA and American Express ***

another keypress. Again, RETURN will terminate the test, and any other key will flop back to let me see the lo-res display again. Line 1190 turns the text display back on.

CONVERT is very straightforward. The outer loop, using the X-register, runs from 23 down to 0. This corresponds to the 24 text lines on the screen, or 48 lo-res rows. If your lo-res picture does not use the bottom 4 lines (8 rows), change line 1300 to "LDX #19".

The inner loop, using the Y-register, runs from 39 down to 0, corresponding to the 40 columns of lo-res pixels. Each of the 960 bytes addressed by X and Y contains two lo-res pixels. The top lo-res row (HLIN 0,39 AT 0) is in the low-order nybble of each of 40 bytes starting at \$400. The second row (HLIN 0,39 AT 1) is in the high-order nybble of the same 40 bytes. The third and fourth rows are in the 40 bytes starting at \$480, and so on. The starting addresses for each row-pair are exactly the same as those for the 24 lines of the text screen. They also happen to be very closely related to the starting addresses for the corresponding rows on the hi-res screen.

I stored the 24 starting addresses in two tables, LOL and LOH. LOL contains the low-half of each address, and LOH the high. Lines 1320-1360 pick up the base address for the current row-pair and put it in pointer LBAS. Lines 1340 and 1370-1380 set up a similar pointer for the hi-res screen. Note that the only difference is that the lo-res screen starts at \$400, and the hi-res screen starts at \$2000. This address points at the first byte (first seven dots) of the top line of the eight hi-res lines that are in the same position as the lo-res row-pair.

Each lo-res pixel will be mapped to four lines on the hi-res screen, and will be seven dots (or one byte) wide. Each of the 960 lo-res bytes has two pixels, so each byte uses eight lines on the hi-res screen. The right lo-res nybble will be the top four lines, and the left nybble will be the bottom four. After studying the tables of hi-res addresses, I noticed that each set of eight lines follow a very regular pattern. Given the address for the leftmost byte of the top line of a set of eight lines, I can compute the addresses for the next seven lines by successively adding 4 to the high byte of the address. Thus the base addresses for the first eight lines are \$2000, \$2400, \$2800, \$2C00, \$3000, \$3400, \$3800, and \$3C00. I can always get the base address for the first of the eight by subtracting \$400 and adding \$2000 to the corresponding lo-res pointer. Line 1370 does that operation in one step with "EOR #\$24".

Lines 1400-1480 pick up the current lo-res byte and feed first the right nybble and then the left nybble to PROCESS.NYBBLE. For indexing purposes I multiply the nybble by 8, so that the lo-res color is in the A-register like this: xCCCCxxx. More on that later. Lines 1490-1530 are the south ends of the two nested loops, equivalent to NEXT Y and NEXT X. By the way, please don't get confused by the terms Y and X. They refer in my program to 6502 registers, not Cartesian coordinates. Just to keep your minds nimble, I use the Y-register for the X-coordinate. The X-register is half the lo-res Y-coordinate.

I mentioned above the problem of coming up with patterns of 28 dots to approximate the lo-res colors. There are only six solid hi-res colors, which correspond exactly to lo-res colors 0, 3, 6, 9, 12, and 15. The other 10 lo-res colors take double the normal hi-res resolution to reproduce exactly. However, as Don Lancaster explains in detail in his "Enhancing Your Apple II -- Volume I", you can produce thousands of shades in hi-res by using dot patterns. I picked 12 of his patterns based on the names he gave them, since I did not have a color monitor. His patterns fit in a 28-dot by two line array. Since each byte stores seven dots, it takes 28 dots before the same of the patterns repeat. Using two lines with different or offset patterns gives even more variety.

The table SHADES in lines 1900-2050 give sixteen patterns. The first four bytes of each color are for the first line of 28 dots, and the other four bytes give the second line of 28 dots. Each lo-res pixel will use only one pair of bytes from the set of eight, depending on which column it is in. The last two bits of the lo-res column number (in the Y-register) select which byte pair to use.

Lines 1650-1700 build an index to the byte pair by merging those two bits with the color*8. Then by addressing "SHADE,X" I get the first byte of a pair, and by using "SHADE+4,X" I get the second one. Each lo-res pixel will use the four hi-res bytes by repeating the pair selected from SHADES.

The rest of the code in PROCESS.NYBBLE involves putting the selected color bytes into the hi-res area. HBAS points at the top line of the four to be stored into, and the Y-register points to the byte on that line; so "STA (HBAS),Y" will store into that byte. COMMON.CODE (so named because of a lack of creativity on my part this morning when I discovered that the same eight lines appeared twice) gets and puts two color bytes. The first byte goes into (HBAS),Y; then I add 4 to the high byte of HBAS (since I KNOW it is zero, ORA can be used to add the bit) and store the second byte at the new (HBAS),Y. The "EOR #\$0C" at line 1720 changes \$24 to \$28 or \$34 to \$38. Similarly, the "EOR #\$1C" at line 1750 changes \$2C to \$30 or \$3C to \$20. This last possibility leaves HBAS prepared for the next column, automatically!

Some of the same tricks could be used in writing a program to copy text from the text screen to the hi-res graphics screen, or for a general purpose routine to write characters onto the hi-res screen. Instead of using a color map, we would need a table of dot-matrix characters. Maybe this is just how everyone does it, but I don't recall seeing all of these tricks in any previous code. Especially the idea of getting the hi-res base pointer by merely toggling two bits in the equivalent text pointer, and the idea of generating successive hi-res pointers by merely adding 4 to that base pointer.

When I wrote this program I wasn't really worrying about speed or space. Nevertheless, as you can see, it is fairly compact. As for speed, it takes less than a second.

[illegible]

One look at the chart will give you some of the reasons there's only one smart choice in 80 column cards for your Apple. But the real secret to Viewmaster 80's success is something even better: Total compatibility.

And the Viewmaster 80 delivers a super sharp, state-of-the-art display with a 7x9 character matrix for clear, easily readable characters. Here are just a few of the powerful features the Viewmaster 80 delivers for a great price (\$139):

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5% sales tax. Add \$10.00 outside U.S.A.

Page 6....Apple Assembly Line.....January, 1986....Copyright (C) S-C SOFTWARE

```

1000 *SAVE S.LORES TO HIRES
1010 *-----
26- 1020 LBAS .EQ $26,27
2A- 1030 HBAS .EQ $2A,2B
2E- 1040 SAVEX .EQ $2E
30- 1050 COLOR .EQ $30
1060 *-----
1070 T
0800- 20 37 09 1080 JSR PLOT
0803- AD 50 C0 1090 LDA $C050 GRAPHICS
0806- AD 52 C0 1100 LDA $C052 SOLID (40 BY 48 PIXELS)
0809- AD 54 C0 1110 LDA $C054 PRIMARY PAGE
080C- AD 56 C0 1120 .1 LDA $C056 LO-RES
080F- 20 23 08 1130 JSR PAUSE.FOR.ANY.KEY
0812- F0 0B 1140 BEQ .2 ...<RETURN>
0814- AD 57 C0 1150 LDA $C057 HIRES
0817- 20 2E 08 1160 JSR CONVERT
081A- 20 23 08 1170 JSR PAUSE.FOR.ANY.KEY
081D- D0 ED 1180 BNE .1 ...NOT <RETURN>
081F- AD 51 C0 1190 .2 LDA $C051 TEXT
0822- 60 1200 RTS
1210 *-----
0823- AD 00 C0 1220 PAUSE.FOR.ANY.KEY
0826- 10 FB 1230 .1 LDA $C000 WAIT FOR ANY KEY
0828- 8D 10 C0 1240 BPL .1 ...NOT YET
082B- C9 8D 1250 STA $C010 CLEAR STROBE
082D- 60 1260 CMP #$8D SET .EQ. IF <RETURN>
1270 RTS
1280 *-----
082E- A2 17 1290 CONVERT
0830- A0 27 1300 LDX #23 OR #19 IF MIXED MODE
0832- BD 73 08 1310 .1 LDY #39 COLUMNS 0...39
0835- 85 26 1320 LDA LOL,X SET UP BASE POINTER FOR LINE
0837- 85 2A 1330 STA LBAS
0839- BD 5B 08 1340 STA HBAS SAME FOR HI-RES
083C- 85 27 1350 LDA LOH,X
083E- 49 24 1360 STA LBAS+1
0840- 85 2B 1370 EOR #$24 SHIFT FROM $400 TO $2000 FOR HI-RES
0842- 86 2E 1380 STA HBAS+1
0844- B1 26 1390 STX SAVEX SAVE X-REG
0846- 48 1400 .2 LDA (LBAS),Y GET TWO LO-RES PIXELS
0847- 0A 1410 PHA SAVE FOR LOWER ONE
0848- 0A 1420 ASL UPPER PIXEL * 8
0849- 0A 1430 ASL
084A- 20 8B 08 1440 ASL
084D- 68 1450 JSR PROCESS.NYBBLE
084E- 4A 1460 PLA GET LOWER PIXEL
084F- 20 8B 08 1470 LSR TIMES 8
0852- 88 1480 JSR PROCESS.NYBBLE
0853- 10 EF 1490 DEY NEXT COLUMN. SCANNING RIGHT TO LEFT
0855- A6 2E 1500 BPL .2 ...ANOTHER ONE
0857- CA 1510 LDX SAVEX RESTORE X-REG
0858- 10 D6 1520 DEX NEXT LINE, SCANNING BOTTOM TO TOP
085A- 60 1530 BPL .1 ...ANOTHER ONE
1540 RTS FINISHED!
1550 *-----
085B- 04 04 05
085E- 05 06 06
0861- 07 07 07 1560 LOH .HS 04.04.05.05.06.06.07.07 HIGH BYTES
0863- 04 04 05
0866- 05 06 06
0869- 07 07 07 1570 .HS 04.04.05.05.06.06.07.07 OF SCRPN PTRS
086B- 04 04 05
086E- 05 06 06
0871- 07 07 07 1580 .HS 04.04.05.05.06.06.07.07 (TEXT OR LO-RES)
1590 *-----
0873- 00 80 00
0876- 80 00 80
0879- 00 80 1600 LOL .HS 00.80.00.80.00.80.00.80 LOW BYTES
087B- 28 A8 28
087E- A8 28 A8
0881- 28 A8 1610 .HS 28.A8.28.A8.28.A8.28.A8 OF SCRPN PTRS
0883- 50 D0 50
0886- D0 50 D0
0889- 50 D0 1620 .HS 50.D0.50.D0.50.D0.50.D0

```

```

1630 #-----
1640 PROCESS.NYBBLE
088B- 29 78 1650 AND #$78 MASK THE SHIFTED NYBBLE
088D- 85 30 1660 STA COLOR
088F- 98 1670 TYA LO-RES COLUMN
0890- 29 03 1680 AND #3 LOW 2 BITS
0892- 05 30 1690 ORA COLOR 000000YY
0894- AA 1700 TAX
0895- 20 A4 08 1710 JSR COMMON.CODE
0898- 49 0C 1720 EOR #$0C 3RD LINE OF 4
089A- 85 2B 1730 STA HBAS+1
089C- 20 A4 08 1740 JSR COMMON.CODE
089F- 49 1C 1750 EOR #$1C NEXT LINE
08A1- 85 2B 1760 STA HBAS+1
08A3- 60 1770 RTS
1780 #-----
1790 COMMON.CODE
08A4- BD B7 08 1800 LDA SHADES.X EVEN LINE
08A7- 91 2A 1810 STA (HBAS),Y
08A9- A5 2B 1820 LDA HBAS+1
08AB- 09 04 1830 ORA #4
08AD- 85 2B 1840 STA HBAS+1
08AF- BD BB 08 1850 LDA SHADES+4,X ODD LINE
08B2- 91 2A 1860 STA (HBAS),Y
08B4- A5 2B 1870 LDA HBAS+1
08B6- 60 1880 RTS
1890 #-----
08B7- 00 00 00
08BA- 00 00 00
08BD- 00 00 1900 SHADES .HS 00.00.00.00.00.00.00.00 0--BLACK
08BF- AA D5 AA
08C2- D5 55 2A
08C5- 55 2A 1910 .HS AA.D5.AA.D5.55.2A.55.2A 1--MAGENTA
08C7- 91 A2 C4
08CA- 88 C4 88
08CD- 91 A2 1920 .HS 91.A2.C4.88.C4.88.91.A2 2--DARK BLUE
08CF- 11 22 44
08D2- 08 44 08
08D5- 11 22 1930 .HS 11.22.44.08.44.08.11.22 3--PURPLE
08D7- 2A 55 2A
08DA- 55 2A 55
08DD- 2A 55 1940 .HS 2A.55.2A.55.2A.55.2A.55 4--DARK GREEN
08DF- 33 66 4C
08E2- 19 4C 19
08E5- 33 66 1950 .HS 33.66.4C.19.4C.19.33.66 5--GRAY 1
08E7- D5 AA D5
08EA- AA D5 AA
08ED- D5 AA 1960 .HS D5.AA.D5.AA.D5.AA.D5.AA 6--MEDIUM BLUE
08EF- DD BB F7
08F2- EE F7 EE
08F5- DD BB 1970 .HS DD.BB.F7.EE.F7.EE.DD.BB 7--LIGHT BLUE
08F7- A2 C4 88
08FA- 91 88 91
08FD- A2 C4 1980 .HS A2.C4.88.91.88.91.A2.C4 8--BROWN
08FF- AA D5 AA
0902- D5 AA D5
0905- AA D5 1990 .HS AA.D5.AA.D5.AA.D5.AA.D5 9--ORANGE
0907- B3 E6 CC
090A- 99 CC 99
090D- B3 E6 2000 .HS B3.E6.CC.99.CC.99.B3.E6 A--GRAY 2
090F- D5 AA D5
0912- AA AA D5
0915- AA D5 2010 .HS D5.AA.D5.AA.AA.D5.AA.D5 B--PINK
0917- 6E 5D 3B
091A- 77 3B 77
091D- 6E 5D 2020 .HS 6E.5D.3B.77.3B.77.6E.5D C--LIGHT GREEN
091F- 2A 55 2A
0922- 55 AA D5
0925- AA D5 2030 .HS 2A.55.2A.55.AA.D5.AA.D5 D--YELLOW
0927- 2A 55 2A
092A- 55 D5 AA
092D- D5 AA 2040 .HS 2A.55.2A.55.D5.AA.D5.AA E--AQUAMARINE
092F- 7F 7F 7F
0932- 7F 7F 7F
0935- 7F 7F 2050 .HS 7F.7F.7F.7F.7F.7F.7F.7F F--WHITE

```


Why Are Apple Owners So Loyal?

People who have the best often are, but in the case of Apple there's more. Apple owners think back to how Apple got started in 1977, just two people working out of a garage and what happened is the talk of Wall Street and the computer industry as well. Many like the fact that Apple only makes computers. Unlike their competition they don't make typewriters, copiers or telephones. They do just one thing and that's one reason they do it so well.

At Applied Engineering we think the same way. You see, Applied Engineering is the only major hardware manufacturer totally dedicated to the Apple computer. Whereas most of our competitors must divide their customer support and engineering time between IBM, Atari, Radio Shack or other computers, our engineers only design products for the Apple. This dedication allows us to be much more familiar with the Apple and the people who use them.

We don't expect you to buy an Applied Engineering peripheral on loyalty alone, but when you compare our products to those made by QUADRAM, MICROSOFT, AST and others you'll find out why Applied Engineering means a quality design, innovation, craftsmanship and total Apple compatibility.

The other guys do pretty well considering how busy they are with IBM. But at Applied Engineering, ALL of our work involves the Apple. In fact, all of our employees were Apple owners before they came to work for us. The people in shipping, engineering, quality control, order entry, all use Apples at work and at home.

This one track mindedness of ours allows us to offer the largest storage with AppleWorks and Supercalc and our Z-80 card now includes the new 4.0 operating system. We can expand the Apple IIe to over 3 MEGABYTES of memory and we've got clock cards, music cards, A to D

converters, digital controllers, and a BSR system so your Apple can control your whole house with no additional wiring!

Applied Engineering recognizes that we've got to do a better job than our IBM counterparts because we know you're smarter than the average computer buyer, you bought an Apple. You see, our competition has it a little easier, their customers aren't as smart as you. After all, they bought the wrong computer.

So if you need more memory, or 80 columns, or RGB color, double hi res graphics, if you want to know the time and temperature or other "real world" conditions, if you'd like to run CP/M software, have a RAM disk, increase the storage of AppleWorks and other programs, if you want your Apple to play music, talk and sing, if you'd like your Apple to control the lights and appliances in your house, then do what NASA does, what Ford does, what the U.S. Government, Hughes Aircraft, Honeywell, Westinghouse, AT&T, Apple Computer, and even what Steve Wozniak does, call Applied Engineering. Then you will discover what thousands of companies and over a hundred thousand Apple owners already know, that you can be smart and loyal all at the same time.

We Set the Standard



APPLIED ENGINEERING

(214) 241-6060

For information and specifications on Applied Engineering's line of Apple peripherals, please see our ads in this magazine. Prices are given.

```

2060 *-----
2070 *   FILL CORNER WITH SAMPLES OF EACH COLOR
2080 *-----
0937- A0 00 2090 PLOT LDY #0
0939- 84 30 2100 STY COLOR
093B- A2 03 2110 LDX #3
093D- A5 30 2120 .2 LDA COLOR 00, 44, 88, CC
093F- 99 00 04 2130 STA $400,Y GR ROWS 0-3
0942- 99 80 04 2140 STA $480,Y
0945- 18 2150 CLC
0946- 69 11 2160 ADC #$11 11, 55, 99, DD
0948- 99 00 05 2170 STA $500,Y GR ROWS 4-7
094B- 99 80 05 2180 STA $580,Y
094E- 69 11 2190 ADC #$11 22, 66, AA, EE
0950- 99 00 06 2200 STA $600,Y GR ROWS 8-11
0953- 99 80 06 2210 STA $680,Y
0956- 69 11 2220 ADC #$11 33, 77, BB, FF
0958- 99 00 07 2230 STA $700,Y GR ROWS 12-15
095B- 99 80 07 2240 STA $780,Y
095E- C8 2250 INY
095F- CA 2260 DEX
0960- 10 DB 2270 BPL .2
0962- 69 11 2280 ADC #$11 ..., 44, 88, CC, END
0964- 85 30 2290 STA COLOR
0966- 90 D3 2300 BCC .1 ...MORE
0968- 60 2310 RTS
2320 *-----

```

A Question About BRUN

Mike Lawrie, a reader in South Africa, reports that he tried our prime benchmark (Sep 85 AAL) in a Titan Accelerator card equipped with a 65802. It ran 1000 times in 41 seconds, which correlates very nicely with my predictions in the article. The Titan card runs at 3.58 MHz, and I predicted .35 seconds for 10 repetitions at 4 MHz.

Mike also asked an interesting question, which has been asked by a lot of you at one time or another. Why is it that some assembly language programs can be BLOAded and CALLed, but not BRUN? Even the following very simple program will not return from a BRUN, while it will from a BLOAD followed by a CALL:

```

JSR $FF3A   Ring the bell
RTS

```

The problem is inside the DOS BRUN command. This command does not use a JSR command to jump to the binary code just loaded. Rather, it uses a JMP command. No return address is left on the stack. When the RTS at the end of the program is executed, it pops garbage off the stack and returns wherever that garbage indicates. What will happen is rather unpredictable.

The Applesoft CALL command does use JSR, and so it works. So does the monitor G command, and the S-C Macro Assembler MGO command. In ProDOS, the BRUN processor works correctly, using a JSR.

This leaves the question: How should a BRUNnable program end under DOS 3.3? If it is to return to the command prompt () for Applesoft) then the last line should be JMP \$3D0. If the BRUN command came from a machine language program (unlikely) then the called program should end with a JMP to a known entry point in the calling program. The most likely case is an Applesoft program that uses a machine language routine. The best way to handle this is to use BLOAD and CALL.

Monthly AAL Source Disk Subscriptions Now Available

We have always made the source code of all the programs published in Apple Assembly Line available on disks. We have collected the programs from three issues together on Quarterly Disks, priced at \$15 each or \$45/year.

Now that diskettes are so much less expensive, we have decided to try another approach. For those who are interested in getting the source code on disk, we would like to send the source disk along with each newsletter. We will still collect three issues onto Quarterly Disks, for late comers. But those of you who have Quarterly Disk subscriptions will start getting the Monthly Disks. We will send the disk and newsletter in the same envelope, First Class Mail.

The price for combined newsletter/disk subscriptions will be \$64 in the USA, Canada, and Mexico. For other countries the postage is higher, so the fee will be \$87.

If you want to synchronize your newsletter and Monthly Disk subscriptions, you can pro-rate the Monthly Disk at \$3.75 per month (\$4.75 for overseas). You can check the length remaining on your current newsletter subscription by looking at the mailing label: the number in the upper-right corner is the year and month of the last issue of your current subscription.

RAMWORKS™

ACCEPT NO SUBSTITUTES. BECAUSE THERE AREN'T ANY.

There's only one card like RamWorks. We've got the best hardware design. We supply the best software and we've got the best support from software companies.

If someone tempts you with an imitation, please get both sides of the story. You'll discover why RamWorks offers the best enhancements to AppleWorks and other programs, and at the lowest price.

GUARANTEED!

**214-241-6060
9 AM - 11 PM**


**"We Set the Standard"
APPLIED ENGINEERING**

Text File Transfer Using DOS 3.3 File Manager.....Bob Potts

Transferring text files from one drive to another can be frustrating and time consuming. The standard procedure is to read from the file on the source drive and write to the file on the target drive. One possible solution is to use FID, but you must BRUN FID and cannot use it from inside an Applesoft program.

With this in mind, I set out to write a utility which will transfer a text file using the DOS 3.3 File Manager (FM) routines. FM has been a part of every release of DOS, but very little documentation has been written about these powerful routines. While RWTS concerns itself with tracks and sectors, FM deals with whole files, be they binary, text, or Applesoft. I recalled that a couple of years ago, Bob Sander-Cederlof had assisted me with a communications program and had used the FM routines to read and store the file. I located the listing we used, analyzed the code, and here is the result.

The entire program could have been written in assembler, but since most of my programs are in Applesoft (with machine language support routines), I decided to write it as simple as possible. The name of the file to be transferred, the OPEN, READ, WRITE, and CLOSE commands are all obtained through a short Applesoft front end program. The machine language portion is broken down as follows.

Lines 1130-1150 are simply easily accessible jump vectors to the two routines which will be CALLED from inside an Applesoft driver.

Lines 1190-1320 clear the buffer, in this case \$2000-95FF, to zeroes. This gives us a buffer of 30,208 bytes, which should be large enough for most text files. (This is 118 sectors.) Lines 1330-1340 reset the base buffer address, for use later to find the end of the data in the buffer.

Lines 1360-1460 load the file that has been OPENed by the Applesoft driver. The process of setting up a FM parameter block is simplified by using a preset data area called RD.BLK, lines 1790-1800. Calling FM.SETUP sets up the Y- and A-registers properly, and then calling FM.ENTRY reads the text file.

Lines 1500-1580 search through the data buffer for the first occurrence of a 00 byte, which will signal the end of data. By subtracting the base buffer address in lines 1660-1710 we get the actual length of the data. Lines 1600-1650 copy in the initial parameter values for writing, and lines 1660-1710 set up the length.

Lines 1720-1740 call on FM to actually write the data on the file that has been opened in the Applesoft driver.

The time saving using this transfer is significant. A text file containing 8000 bytes took 49 seconds to read and write using pure Applesoft. Using the FM the same operation was

accomplished in only 17 seconds.

Since the program is only 120 bytes long, it can be placed almost anywhere there is free space, especially on page 3. If you are working from a larger Applesoft program, the starting point for the buffer could be moved as needed to load your text file.

```

2000- 1000 *SAVE POTTS TEXT COPIER
      1010 *-----
      1020          .OR $300
      1025          .IF TEXT.TRANSFER.OBJ
      1030 *-----
      1040 MY.BUFFER .EQ $2000
      1050 *-----
      E0- 1060 BUFFER      .EQ $E0,E1      POINT TO FILE BUFFER
      E2- 1070 RESULT     .EQ $E2        FILE MANAGER RETURN CODE
      1080 *-----
      03DC- 1090 FM.SETUP   .EQ $3DC      INITIALIZE Y & A
      03D6- 1100 FM.ENTRY   .EQ $3D6      FILE MANAGER ENTRY POINT
      B5BB- 1110 FM.BLK     .EQ $B5BB     FILE MANAGER FARM LIST
      1120 *-----
      0300- 1130          SET UP JUMP VECTORS
      4C 06 03 1140          JMP INITIALIZE.AND.READ
      0303- 4C 3A 03 1150          JMP FIND.END.AND.WRITE
      1160 *-----
      1170 INITIALIZE.AND.READ
      1180 *-----
      1190 INITIALIZE.THE.BUFFER
      0306- A9 00 1200          LDA #MY.BUFFER
      0308- 85 E0 1210          STA BUFFER      LSB
      030A- A9 20 1220          LDA /MY.BUFFER
      030C- 85 E1 1230          STA BUFFER+1      MSB
      030E- A0 00 1240          LDY #0
      0310- A9 00 1250          .1 LDA #0          CLEAR BUFFER UP TO $95FF
      0312- 91 E0 1260          .2 STA (BUFFER),Y
      0314- C8      1270          INY
      0315- D0 FB 1280          BNE .2          NEXT BYTE IN THIS PAGE
      0317- E6 E1 1290          INC BUFFER+1    ...STILL IN THE PAGE
      0319- A5 E1 1300          LDA BUFFER+1    NEXT PAGE
      031B- C9 96 1310          CMP #$96
      031D- D0 F1 1320          BNE .1          AT END OF STORAGE?
      031F- A9 20 1330          LDA /MY.BUFFER  ...NO, KEEP CLEARING
      0321- 85 E1 1340          STA BUFFER+1    RESET BUFFER POINTER
      1350 *-----
      1360 READ.THE.FILE
      0323- A2 09 1370          LDX #9          10 BYTES
      0325- BD 6F 03 1380          .1 LDA RD.BLK,X
      0328- 9D BB B5 1390          STA FM.BLK,X
      032B- CA      1400          DEX
      032C- 10 F7 1410          BPL .1
      032E- 20 DC 03 1420          JSR FM.SETUP
      0331- 20 D6 03 1430          JSR FM.ENTRY
      0334- AD C5 B5 1440          LDA FM.BLK+10   GET RETURN CODE
      0337- 85 E2 1450          STA RESULT     SAVE FOR APPLESOFT PEEK
      0339- 60      1460          RTS          RETURN TO APPLESOFT
      1470 *-----
      1480 FIND.END.AND.WRITE
      1490 *-----
      1500 FIND.END.OF.BUFFER
      033A- A0 00 1510          LDY #0          SEARCH FOR 00 BYTE
      033C- B1 E0 1520          .1 LDA (BUFFER),Y
      033E- F0 07 1530          BEQ .2          ...FOUND END
      0340- C8      1540          INY
      0341- D0 F9 1550          BNE .1          ...NEXT BYTE IN SAME PAGE
      0343- E6 E1 1560          INC BUFFER+1    NEXT PAGE
      0345- D0 F5 1570          BNE .1          ...ALWAYS
      0347- 84 E0 1580          .2 STY BUFFER    LSB OF EOF BYTE
      1590 *-----
      1600 WRITE.FILE
      0349- A2 09 1610          LDX #9          10 BYTES
      034B- BD 79 03 1620          .1 LDA WR.BLK,X
      034E- 9D BB B5 1630          STA FM.BLK,X
      0351- CA      1640          DEX
      0352- 10 F7 1650          BPL .1
      0354- A5 E0 1660          LDA BUFFER      LSB
      0356- 8D C1 B5 1670          STA FM.BLK+6  LSB OF FILE LENGTH

```

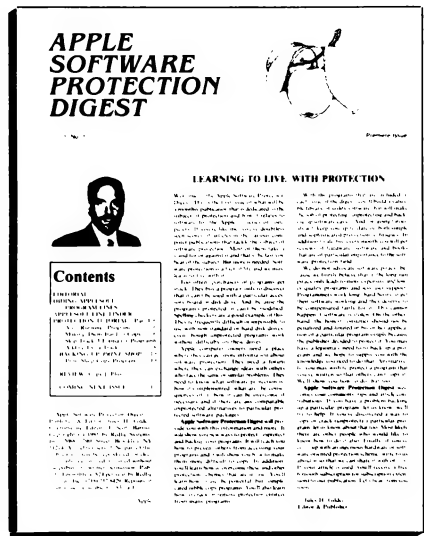
SOFTWARE PROTECTION TECHNIQUES EXPOSED!

Now, for the first time, owners of Apple // series computers can learn all about the tricks and techniques used to protect Apple software. Apple Software Protection Digest, a new monthly publication, will show you how to protect, unprotect and backup your software.

- Prevent others from accessing your programs
- Make your programs difficult to copy
- Overcome protection schemes on commercial software
- Build a library of protection-oriented utility programs
- Get help with your specific problems
- Learn about the latest advances in protection hardware and software

All this and more can be yours by subscribing to the Apple Software Protection Digest. A one-year subscription is \$24, two years is \$42.

SUBSCRIBE TODAY!



REDLIG SYSTEMS, INC., Dept. A135
2088 - 79th St., Brooklyn, NY 11214

Please enter my _____ year subscription to Apple Software Protection Digest.

☐ Enclosed is my check for _____

☐ Please charge my credit card: ☐ VISA ☐ MasterCard ☐ American Express

Card Number _____ Exp. Date _____ Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

```

0359- 38      1680      SEC
035A- A5 E1    1690      LDA BUFFER+1
035C- E9 20    1700      SBC /MY.BUFFER
035E- 8D C2 B5 1710      STA FM.BLK+7      MSB OF FILE LENGTH
0361- 20 DC 03 1720      JSR FM.SETUP
0364- A2 01    1730      LDX #1      IF NO FILE, ALLOCATE ONE
0366- 20 D6 03 1740      JSR FM.ENTRY    WRITE THE FILE
0369- AD C5 B5 1750      LDA FM.BLK+10    RETURN CODE
036C- 85 E2    1760      STA RESULT      SAVE FOR APPLESOFT PEEK
036E- 60      1770      RTS      RETURN TO APPLESOFT
      1780 *-----*
036F- 03 02 00
0372- 00 00 00 1790 RD.BLK .HS 03.02.0000.0000
0375- 00 76 00
0378- 20      1800      .DA $9600-MY.BUFFER,MY.BUFFER
0379- 04 02 00
037C- 00 00 00 1810 WR.BLK .HS 04.02.0000.0000
037E- 00 76 00
0382- 20      1820      .DA $9600-MY.BUFFER,MY.BUFFER
      1830 *-----*

```

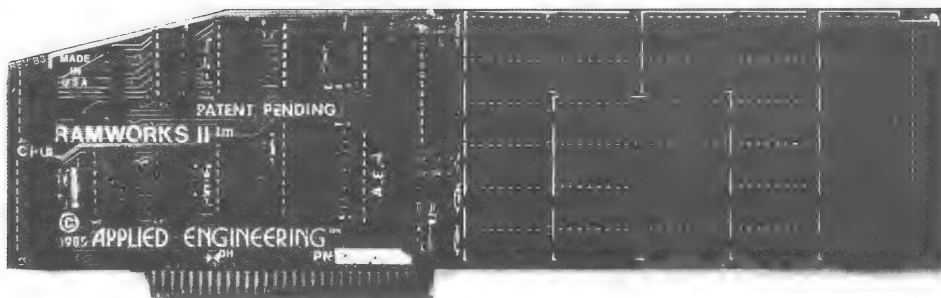
```

10  REM  PROGRAM TO TRANSFER A TEXT FILE
20  REM  FROM DRIVE 1 TO DRIVE 2 USING
30  REM  DOS 3.3 FILE MANAGER ROUTINES
40  REM  ---BY BOB POTTS. LOUISVILLE, KENTUCKY---
99  REM  -----
100 HIMEM: 8192: REM  $2000-95FF IS MY BUFFER
110 D$ = CHR$ (4)
120 PRINT D$"NOMON I,O,C"
130 PRINT D$"BLOAD TEXT.TRANSFER.OBJ"
199 REM -----
200 RF = 768: REM CALL ADDRESS TO READ FILE
210 WF = 771: REM CALL ADDRESS TO WRITE FILE
220 RC = 226: REM PEEK ADDRESS FOR FM RETURN CODE
300 REM -----
310 TEXT : HOME : PRINT "TEXT FILE TRANSFER"
320 PRINT "-----"
330 INPUT "ENTER FILE NAME: ";F$
400 REM READ FILE FROM DRIVE 1
410 PRINT D$"OPEN"F$,D1
420 PRINT D$"READ"F$
430 CALL RF
440 IF PEEK (RC) = 5 THEN 500
450 PRINT D$"CLOSE"
460 PRINT "RETURN CODE NOT 'END OF DATA'"
470 STOP
500 REM WRITE FILE TO DRIVE 2
510 PRINT D$"CLOSE"
520 PRINT D$"OPEN"F$,D2
530 PRINT D$"WRITE"F$
540 CALL WF
550 IF PEEK (RC) = 0 THEN 600
560 PRINT D$"CLOSE"
570 PRINT "RETURN CODE WAS " PEEK (RC)
580 STOP
600 REM FINISHED
610 PRINT D$"CLOSE"
620 PRINT "TRANSFER COMPLETE"
630 END

```

Meet RamWorks II[®]

The Recognized Industry Standard For Memory Expansion of the Apple IIe.



RamWorks II. A Generation Ahead. Again.

The best selling expansion card for the Apple IIe just got even better. With RamWorks II, expand your IIe to an incredible 3 megabytes of usable RAM.

Turbo Charged AppleWorks.

RamWorks II plugs into the IIe auxiliary slot and acts just like Apple's extended 80 column card, only better—because if you buy a 256K or larger card, AppleWorks will automatically load itself into RamWorks II. This dramatically increases AppleWorks' speed and power because it effectively eliminates the time required to access disk drive 1. Now, switch from word processing to spreadsheet to database management at the speed of light. AppleWorks responds the moment your fingers touch the keyboard.

But AppleWorks has certain internal limits, independent of available memory. Fear not. Only RamWorks II (and the original RamWorks of course) removes those limits. Only RamWorks II increases

the maximum number of records available from 1,350 to over 16,000. Only RamWorks II actually increases the number of lines permitted in the word processing mode. And only RamWorks II features a built-in printer buffer, so you no longer have to wait for your printer to stop before going back to AppleWorks (256K or larger RamWorks II required).

With RamWorks II, you won't have to split your data into 2 or more separate files because you'll have the necessary memory to access ALL your data ALL the time, quickly and conveniently.

RamWorks II	AppleWorks Desktop
128K	101K
256K	188K
512K	378K
1 MEG	758K
1.5 MEG	1136K
3 MEG	2277K

The Most Friendly, Most Expandable Card Available.

RamWorks II is compatible with more off-the-shelf software than any other RAM card. Popular programs like Advanced VisiCalc, Magic Office System, FlashCalc, The Spread Sheet, Diversi-DOS, Supercalc 3A, MagiCalc, etc. (and hardware add-ons like Profile and Sider hard disks). Fact is, only RamWorks is 100% compatible with all software written for the Apple 80 column and extended 80 column cards. In addition, RamWorks II can emulate most other RAM cards, so you can use programs written for them without modification. And any size RamWorks II can be user upgraded later to any larger size.

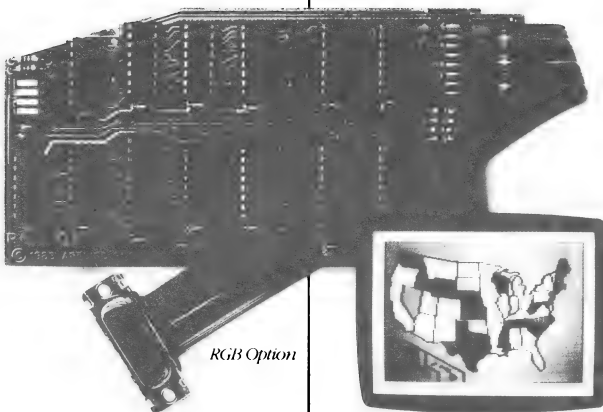
RamWorks II was designed so you could take full advantage of future developments in 16 and 32 bit microprocessors. As your needs grow, so can RamWorks II. A handy coprocessor connector allows the latest and greatest coprocessor cards to access all 3 MEG

of RamWorks II memory. And speaking of more memory, RamWorks II has a memory expansion connector on board so a low profile (no slot 1 interference) memory expansion card can add another 512K or 2 MEG of memory.

Unlike Apple's smaller, more expensive RAM card, RamWorks II plugs into the IIe auxiliary slot and therefore leaves slots 4 and 5 available for other peripheral cards.

It's In Color

RamWorks II by itself is *fully* compatible with both the Apple monochrome and color monitors. But if you want better color graphics *plus* a more readable 80 column text (that blows away any composite color monitor) you'll appreciate our RGB color option. For only \$129, it can be added to RamWorks II, giving you a razor sharp, vivid brilliance that's unsurpassed in the industry. The RGB option does not waste another valuable slot, but rather plugs into the back of RamWorks II with no slot 1 interference (works on the original RamWorks, too) and attaches to virtually any RGB monitor. And remember: You can order



the RGB option with your RamWorks II. Or add it on at a later date.

It Corrects Mistakes.

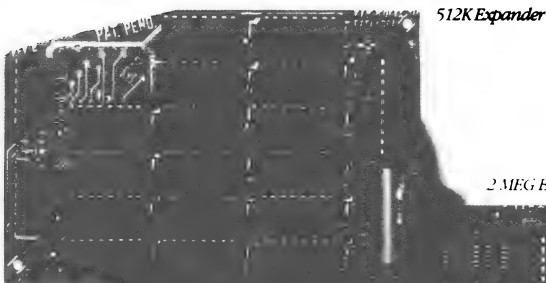
Let's say you bought some other RAM card (and that's a mistake) and your RAM card is not being recognized by AppleWorks, Advanced Visicalc, Flashcalc, Supercalc 3A, or other programs, and you want RamWorks II.

No problem. The memory chips on the card that you now have, which is where most of the money is, can be unplugged and then plugged into the expansion sockets on RamWorks II.

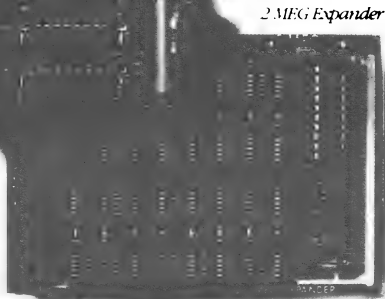
It's Got It All.

- 15 Day Money Back Guarantee
- Super sharp 80 column text (patent pending) with or without RGB option
- Double high resolution graphics (with or without RGB option)
- Expandable up to 1 Meg (1024K) on main board

- Expandable to 3 Meg (3072K) with expander (piggyback) card
- Can use 64K or 256K RAMs in any combination
- Linear addressing coprocessor port
- Automatic AppleWorks expansion up to 2277K desktop
- Accelerates AppleWorks
- Built-in AppleWorks printer buffer
- The only large RAM card that's 100% compatible with all IIe software



512K Expander



2 MEG Expander

- RamDrive™ the ultimate disk emulation software included free
- 16 Bit option
- Compatible RGB option
- Built-in self diagnostics software
- No slot 1 interference
- Lowest power consumption (patent pending)
- Takes only one slot (auxiliary)
- Software industry standard
- Advanced Computer Aided Design
- Used by Apple Computer, Steve Wozniak and virtually all software companies
- Displays date and time on the AppleWorks screen with any PRO-DOS compatible clock
- 5 Year no hassle warranty

RamWorks II with 64K	\$ 179
RamWorks II with 256K	\$ 219
RamWorks II with 512K	\$ 269
RamWorks II with 1 MEG	\$ 389
RamWorks II with 1.5 MEG	\$ 549
RamWorks II with 3 MEG	\$1699
RGB Option (may add later)	\$ 129
16 Bit Option (may add later)	\$ 89

RamWorks II. The industry standard for memory expansion of the Apple IIe.

ORDER YOUR RamWorks II TODAY: 9 a.m. to 11 p.m. 7 days, or send check or money order to Applied Engineering. MasterCard, Visa and C.O.D. welcome. Texas Residents add 5% sales tax. Add \$10.00 outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Fast 6502 & 65802 Multiply Routines.....Bob Sander-Cederlof

Since multiplication is not a built-in function in the 6502, 65C02, or 65802, many of us have written our own subroutines for the purpose. I will present some efficient subroutines here, to handle the 8-bit and 16-bit cases.

I will assume both arguments are the same length (either 8-bits or 16-bits) and that we want the full product. If the arguments are only 8-bits long, the product will be 16-bits long. If the arguments are 16-bits long, the product will be 32-bits long. I will also assume the arguments are unsigned values. Thus \$FF times \$FF will be \$FE01 (in decimal, 255x255 = 65025).

Way back in February, 1981, I published an article with a Brooke Boering's fast 16-bit multiplication subroutine. His subroutine duplicated the functions of the subroutine in the original Apple Monitor ROM, but was nearly twice as fast. Brooke's programs were originally published in the December, 1980, Micro magazine (now defunct). He included an 8-bit multiply subroutine with an average time of only 192 cycles.

Damon Slye wrote an article for Call APPLE, published June, 1983. He introduced some coding tricks which allow an 8-bit multiply in an average of 160 cycles. I have reproduced Damon's program below, in lines 1010-1300. His trick involves eliminating a CLC opcode from the loop in lines 1210-1260. Ordinarily you would need a CLC before the ADC instruction; Damon decremented the multiplicand by one before starting the loop, so that adding with carry set works. He does the decrementing in lines 1130-1160. Note that if the original multiplicand was zero, he skips all the rest of the code and just returns the answer: 0.

I had to go at least one step faster, so I partially "un-wrapped" the 8-step loop. I changed it to loop only four times, but handled two bits of the multiplier each time. This runs an average time of 140 cycles. You could unwrap it all the way, writing out the BCC-ADC-ROR-ROR lines a total of 8 times, and cut the average time down to only 111 cycles.

Let me stop here and say what I mean by average time. I am stating time in terms of "cycles", rather than seconds or microseconds. The Apple two different cycle times, depending on the video timing logic. The average Apple speed is 1020488 cycles per second. The multiply algorithms will vary in speed depending on the number of bits in the multiplier which equal "1". If the multiplier = \$FF (all ones) the algorithm will take the maximum time. If the multiplier is \$00, it will take the minimum time. On the average for random arguments, the multiplier will have four zeroes and four ones, so the average time is equal to the average of the minimum and maximum times. For all of the subroutines, I included the cycles for a JSR to call them, and for the RTS at the end.

I programmed an 8-bit multiply using 65802 opcodes, as shown below in lines 1560-1790. The program is slightly shorter (one

byte), but that really isn't a fair comparison. The arguments and product are handled differently, and so the effort to call the program may be more or less than that for the 6502 version. Rather than passing the multiplicand in the X-register, I have it in the A-register. I pass the multiplier in the high byte of the A-register. Since X is not used for passing any values, I saved and restored it (lines 1620 and 1770). I assumed the program would be called from the 6502 mode, which of course it was as long as I was testing it. In "real life" it might be written to be called from Native 65802 mode, since the larger program it was a part of would also be taking advantage of all the 65802 features.

I used a couple of tricks to save space and time. One you may justly complain about is that I store the multiplicand directly into the operand field of the ADC instruction at line 1720. This definitely saves time, but it also could have serious drawbacks. (For example, it would not work if executed from ROM.) Since I enter in 6502 Emulation mode, line 1640 only loads 0 into the low byte of the A-register. Lines 1650-1660 enter the 65802 Native mode. Line 1680 sets the A-register to 16-bit mode.

In line 1690 I form the inverse (one's complement) of the multiplier. This is just another way of eliminating the CLC from the loop. Note that the multiplier is in the high byte of A, and the product is going to be accumulated in the low byte. The loop runs from line 1700 through line 1740. Line 1700 shifts to the left both the partial product and what remains of the multiplier, putting the highest remaining bit of the multiplier into the carry status bit. If that bit = 1, then the original bit in the multiplier before complementing was a zero, so we do not add the multiplicand to the current partial product. As we continue through the loop, the bits of the multiplier keep shifting out just ahead of the ever-growing partial product, until finally we have the answer.

Lines 1750-1780 restore the machine state to the 6502 Emulation mode and restore the original X-register value. The full product is now in the A-register. If I wanted to print out the product, I might do it like this:

```

XBA          GET HIGH BYTE INTO LOW-A
JSR $FDDA    MONITOR PRINT-BYTE SUBROUTINE
XBA          GET LOW BYTE INTO LOW-A
JMP $FDDA

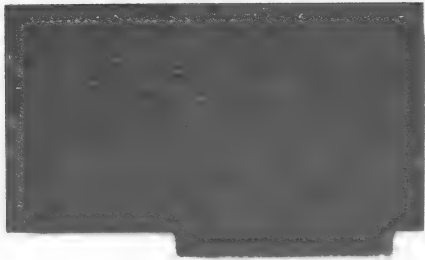
```

Here is a summary of the execution times (in cycles) for the three 8-bit multiply subroutines:

	Minimum	Maximum	Average
Slye	152	168	160
RBSC	132	148	140
65802	119	135	127

The 65802 version would be seven cycles faster if we did not require saving and restoring the X-register. If you want to change the 65802 version for calling from Native mode, delete

With Z-80 Plus,TM run CP/M[®]—the largest body of software in existence.



*Now, get two computers in one,
and all the advantages of both.*

Enter the CP/M world with the new Z-80 Plus card from Applied Engineering, and introduce your Apple II[®] or II⁺ to the thousands of CP/M programs. Only the Z-80 Plus comes standard with the new 4.0 software, the most advanced system ever for running CP/M programs.

The new 4.0 boasts advanced features like built-in disk emulation for popular memory expansion boards, boosting both system speed and storage capacity. And menu-driven utilities that let you get to work faster. The Z-80 Plus also lets you run older CP/M programs — all the way down to Version 1.6 (2.2 is the most popular).

The Z-80 Plus is the only card on the market capable of accessing more than 64K in an Apple IIe. If you have an extended 80-column card, all 128K is usable, and if you have RamWorks, up to 1088K is available.

Each Z-80 Plus comes with our CP/M Ram Drive software, enabling IIe owners to use an extended 80-column card or a RamWorks card as a high-speed Ram disk which runs CP/M software up to *twenty times faster*. So packages like WordStar and dBASE II run at blinding speed.

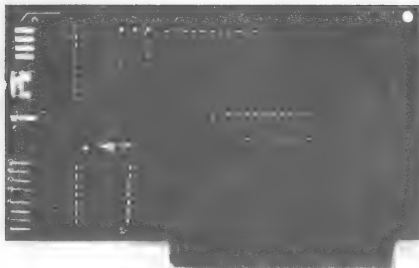
Simply plug the Z-80 Plus into any slot in your Apple. You'll get the benefits of two computers in one — all at an unbelievably low price (only \$139!).

- Fully compatible with ALL CP/M software
- Fully compatible with most hard disks, including Corvus and the Sider
- Fully compatible with Microsoft disks (no pre-boot required)
- Specifically designed for high speed operation in the Apple IIe (runs just as fast in the Apple II+ and Franklin)
- Runs WordStar, dBASE II, Turbo Pascal, Fortran-80, Peachtree and ALL other CP/M software with *no pre-boot*
- Semi-custom I.C. and low parts count allows Z-80 Plus to fly through CP/M programs with extremely low power consumption (we use the Z-80B)
- Does EVERYTHING other Z-80 boards do, *plus* Z-80 interrupts
- Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering, MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Timemaster H.O.TM, the only clock that displays time and date on AppleWorksTM screens and files.



*Now, get all the features of
all the competition combined!*

It's the smart way to put the time and date on your Apple II⁺ or IIe[®]. Because only the Timemaster H.O. packs *ALL* the features of all the competition *combined*, including leap year, year (not just in PRO-DOS), month, date, day of week, hours, minutes, seconds and milliseconds. It's totally PRO-DOS, DOS 3.3, PASCAL and CP/M compatible. And of course, it works better than any other clock with AppleWorks.

If you're using or writing software for other clock cards, you're still covered. Because the H.O. will *automatically* emulate them. And the Timemaster H.O. adds 14 new commands to BASIC. The H.O. even comes complete with two disks full of sample programs, including a computerized appointment book, a DOS dating program, interrupt programs, and over 50 programs that others charge extra for — or don't even offer.

As a low-cost option, you can add true BSR remote control to the H.O., giving you remote control of up to 16 lights and appliances in your home or office.

- Fully PRO-DOS and DOS 3.3, CP/M and PASCAL compatible
- Time in hours, minutes, seconds and milliseconds (the *ONLY* PRO-DOS compatible card with millisecond capability), date with year, month, day of week and leap year
- 24-Hour military format or 12-hour AM/PM format
- Eight software controlled interrupts so you can run two programs at the same time (many examples included)
- Allows AppleWorks to time and date stamp all data automatically
- The only clock card that displays time and date on the AppleWorks screen
- Five year warranty

Clock price \$129.00
BSR option (may be added later) \$ 49.00

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering, MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

lines 1650, 1660, 1750, and 1760. Then insert the following:

```

1612      PHP
1614      SEP #30
...
1772      PLP

```

These changes add one cycle to the time.

```

02-      1000 *SAVE S.MULTIPLY 8X8
03-      1010 *-----
04-      1020 CAND .EQ 2
      1030 PLIER .EQ 3
      1040 PROD .EQ 4,5
      1050 *-----
      1060 * FAST 6502 MULTIPLICATION, BY DAMON SLYE
      1070 * CALL APPLE, JUNE 1983. P45-48.
      1080 * (A-REG) = MULTIPLIER
      1090 * (X-REG) = MULTIPLICAND
      1100 * RETURNS PRODUCT IN A,X (X=LO-BYTE)
      1110 *-----
      1120 FAST.8X8.SLYE
0800- E0 00      1130 CPX #0
0802- F0 17      1140 BEQ .3          A*0=0
0804- CA          1150 DEX          DECR. CAND TO AVOID
0805- 86 02      1160 STX CAND          THE CLC BEFORE ADC CAND
0807- 4A          1170 LSR          PREPARE FIRST BIT
0808- 85 03      1180 STA PLIER
080A- A9 00      1190 LDA #0
080C- A2 08      1200 LDX #8
080E- 90 02      1210 .1 BCC .2          NO ADD
0810- 65 02      1220 ADC CAND
0812- 6A          1230 .2 ROR
0813- 66 03      1240 ROR PLIER
0815- CA          1250 DEX
0816- D0 F6      1260 BNE .1
0818- A6 03      1270 LDX PLIER
081A- 60          1280 RTS
081B- 8A          1290 .3 TXA
081C- 60          1300 RTS
      1310 *-----
081D- E0 00      1320 FAST.8X8.RBSC
081F- F0 1E      1330 CPX #0
0821- CA          1340 BEQ .3          A*0=0
0822- 86 02      1350 DEX          DECR. CAND TO AVOID
0824- 4A          1360 STX CAND          THE CLC BEFORE ADC CAND
0825- 85 03      1370 LSR          PREPARE FIRST BIT
0827- A9 00      1380 STA PLIER
0829- A2 04      1390 LDA #0
082B- 90 02      1400 .1 BCC .2          NO ADD
082D- 65 02      1420 ADC CAND
082F- 6A          1430 .2 ROR
0830- 66 03      1440 ROR PLIER
0832- 90 02      1450 BCC .25          NO ADD
0834- 65 02      1460 ADC CAND
0836- 6A          1470 .25 ROR
0837- 66 03      1480 ROR PLIER
0839- CA          1490 DEX
083A- D0 EF      1500 BNE .1
083C- A6 03      1510 LDX PLIER
083E- 60          1520 RTS
083F- 8A          1530 .3 TXA
0840- 60          1540 RTS
      1550 *-----
      1560 .OP 65816
      1570 *-----
      1580 * MULTIPLIER IN A(15-8), MULTIPLICAND IN A(7-0)
      1590 * RETURN PRODUCT IN A(15-0)
      1600 *-----
      1610 MULTIPLY.8X8.65802
000841- DA          1620 PHX
000842- 8D 54 08    1630 STA .2+1      SAVE MULTIPLICAND
000845- A9 00      1640 LDA #0

```

000847-	18	1650	CLC	
000848-	FB	1660	XCE	
000849-	A2 08	1670	LDX #8	
00084B-	C2 20	1680	REP #\$20	A-REG 16 BITS
00084D-	49 00 FF	1690	EOR ##\$FF00	COMPLEMENT MULTIPLIER
000850-	0A	1700 .1	ASL	
000851-	B0 03	1710	BCS .3	...IF ORIGINAL BIT=0
000853-	69 00 00	1720 .2	ADC #0	ADD MULTIPLICAND
000856-	CA	1730 .3	DEX	
000857-	D0 F7	1740	BNE .1	
000859-	38	1750	SEC	
00085A-	FB	1760	XCE	
00085B-	FA	1770	PLX	
00085C-	60	1780	RTS	
		1790	*-----	

I will also show three sample 16-bit multiply subroutines....no, four. The first one is a copy of Brooke Boering's code. The second is a direct conversion of Brooke's code to 65802 code, with emphasis on space. The third modifies the second with the tricks of Damon Slye; it takes more space, but it is faster.

The first three of these subroutines are modeled after the code in the original Apple monitor ROM. The arguments are expected in page zero locations, low-byte first. The result will also be in page zero locations. The function performed is actually a little more than just multiplication, because it is possible to specify an addend as well. The final result will be $PRODUCT = ADDEND + (MULTIPLIER * MULTIPLICAND)$. $PRODUCT$ is stored in four consecutive bytes, backwards. The highest byte is at $PRODUCT+1$, the next at $PRODUCT$, the next at $PLIER+1$, and the lowest at $PLIER$. The fourth subroutine differs in that the product does not overlap the multiplier.

Looking at Brooke's version (lines 1000-1270) you can see that the loop contains a 16-bit addition (lines 1130-1190). There are also two 16-bit ROR shifts, at lines 1200-1230. These are the likely candidates for shortening via 65802 code. My first version for the 65802 made no other changes in the loop. I merely prefixed Brooke's code with $CLC-XCE-REP$ to get into the 16-bit Native mode, and suffixed it with $SEC-XCE$ to get back to Emulation mode. Then I noticed another shortcut, and the result is in lines 1300-1480.

By moving the $LDA PRODUCT$ up before the BCC opcode in lines 1370-1380, I was able to change a $ROR PRODUCT$ to a simple ROR on the A-register followed by a $STA PRODUCT$. This saves a net six cycles when the multiplier bit is "1", and costs two cycles when the multiplier bit is "0". The average savings for random multipliers is four cycles, inside a loop that runs 16 times.

The faster version, in lines 1500-1780, merely implements Damon Slye's trick of pre-decrementing the multiplicand so as to avoid an explicit CLC opcode inside the 16-time loop. It costs 12 cycles for the extra setup, but it saves two cycles for each one-bit in the multiplier.

The fourth version, in the separate listing as lines 1000-1430, uses the trick of splitting the multiplier in half. In effect,

APPLIED ENGINEERING INTRODUCES THE TRANSWARP™ ACCELERATOR

The Fastest Apple Accelerator Available

TransWarp is the new accelerator card from Applied Engineering. With a TransWarp card in your Apple II, II+, or IIfx, all software will run up to 3.6 times faster (3.1 times faster is average). TransWarp works with all Apple software including AppleWorks, SuperCalc 3a, VisiCalc, and all educational software, graphics and games. TransWarp is compatible with all standard peripheral cards such as Ramworks II and Apple memory cards. Profile and Sider hard disks, 3 1/4" UniDisk, 80 column cards, modems, clock cards, mouses and more. You name it, TransWarp accelerates it! Should 16 bit software become available for the Apple, you can get a low cost 16 bit upgrade chip at any time.

Why TransWarp Is Best

The other speedup cards only speedup your Apple's main memory but TransWarp has 256K of ultra-fast RAM that accelerates your Apple's main memory, ROM memory and auxiliary memory. And since more and more programs are residing in auxiliary memory, TransWarp can run these programs 3 times faster than the competition. TransWarp doesn't use memory caching, which means TransWarp accelerates all software.

It Couldn't Be Easier

Just plug TransWarp into any slot in your Apple II, II+, or IIfx, including slot 3 in the IIfx. Just turn on your Apple and zoom off, it runs 3 1/2 times faster. Should you ever want to run at normal speed, just press the ESC key while turning on your Apple. No pre-boot disks are needed. In fact, no software comes with TransWarp because none is needed as the TransWarp acceleration is completely transparent.

All for only \$279.

- 3.6 MHz 65C02
- 256K of ultra-fast on board RAM
- Accelerates main and auxiliary memory
- Low power design for cool operation
- Totally transparent operation with all software
- Plugs into any slot including slot 3 of an Apple IIfx
- Accelerated 16 bit option available

TransWarp will make you more productive (3 1/2 times more productive, in fact) because your spreadsheets will recalculate faster, and your word processor will move text in the blink of an eye. Accounting, engineering, educational, tax analysis, and even games will all have warp speed.

With a TransWarp card in your Apple, you'll easily pass by others using IBM PCs and ATs. In fact, you'll have one of the fastest computers in the galaxy and although TransWarp leaves the competition in the dust, it's still priced far lower, at only \$279.

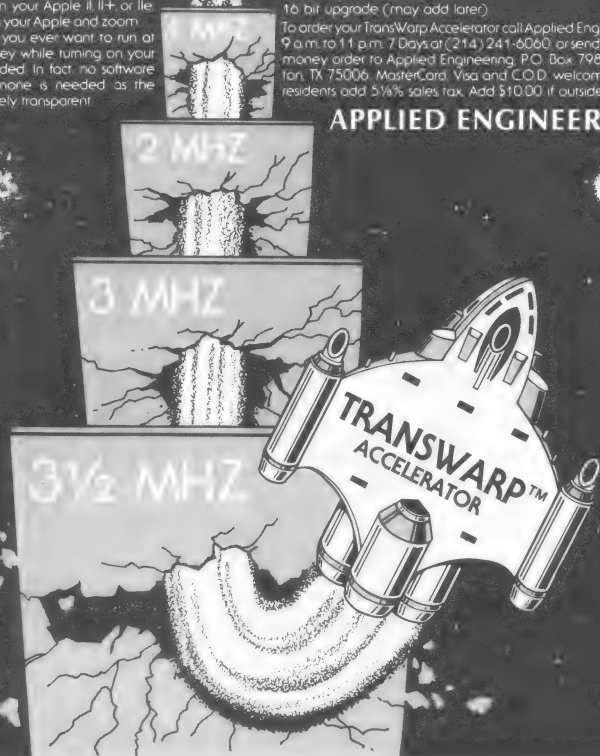
So the next time you're waiting for your computer to finish, give Applied Engineering a call and we'll beam you a TransWarp accelerator card. With our risk free 15 day money back guarantee, you have nothing to lose, except a lot of wasted time.

TransWarp™ Accelerator
16 bit upgrade (may add later)

\$279
\$300

To order your TransWarp Accelerator call Applied Engineering, 9 a.m. to 11 p.m. 7 days at (214) 241-6060, or send check or money order to Applied Engineering, P.O. Box 798, Carrollton, TX 75006. MasterCard, Visa and C.O.D. welcome. Texas residents add 5 1/8% sales tax. Add \$10.00 if outside U.S.A.

APPLIED ENGINEERING



two parallel 8-bit by 16-bit multiplies are accomplished, with the result usually taking less time than any of the other algorithms. By deleting line 1130 (which shaves off another four cycles) the feature of allowing an addend can be included.

Here is a summary of the execution cycles for the four 16-bit multiply subroutines:

	Minimum	Maximum	Average
Boering	541	845	693
Smaller	519	599	559
Faster	531	579	555
Fourth	332	684	508 (usually fastest)

Note that the third subroutine also goes even faster when the multiplicand = zero, because the bulk of the code is skipped.

These are pretty good subroutines, but I have no doubt that they can be improved upon. Why not try your hand? If you can significantly improve either space or time or features, send your code to AAL. We'll publish the best ones, and help advance the state of the art. And if you have some classy division subroutines, they are welcome too!

```

1000 *SAVE S.MULTIPLY 16X16
1010 *-----
00- 1020 PLICAND .EQ $00,01    MULTIPLICAND
02- 1030 PLIER .EQ $02,03    MULTIPLIER, LO-16 OF PRODUCT
04- 1040 PRODUCT .EQ $04,05  HI-16 OF PRODUCT
1050 *-----
1060 .OP 6502
1070 *-----
1080 MULTIPLY.16X16.6502
1090 LDX #16
0800- A2 10 1100 .1 LDA PLIER    CHECK NEXT BIT OF MULTIPLIER
0802- A5 02 1110 LSR
0804- 4A 1110 BCC .2        ...DON'T ADD MULTIPLICAND
0805- 90 0D 1120 CLC
0807- 18 1130 LDA PRODUCT
0808- A5 04 1140 ADC PLICAND
080A- 65 00 1150 STA PRODUCT
080C- 85 04 1160 LDA PRODUCT+1
080E- A5 05 1170 ADC PLICAND+1
0810- 65 01 1180 STA PRODUCT+1
0812- 85 05 1190 ROR PRODUCT
0814- 66 05 1200 .2 ROR PLIER+1
0816- 66 04 1210 ROR PRODUCT
0818- 66 03 1220 ROR PLIER
081A- 66 02 1230 DEX
081C- CA 1240 BNE .1
081D- D0 E3 1250 RTS
081F- 60 1260 *-----
1270 .OP 65802
1280 *-----
1290 MULTIPLY.16X16.65802.SMALLER
1300 CLC
000820- 18 1310 XCE
000821- FB 1320 REP #20    NATIVE MODE
000822- C2 20 1330 LDX #16    A-REG 16-BITS
000824- A2 10 1340 LDA PLIER    LOOP 16 TIMES
000826- A5 02 1350 .1 LDA PLIER    CHECK NEXT BIT OF MULTIPLIER
000828- 4A 1360 LSR
000829- A5 04 1370 LDA PRODUCT    GET HI-16 OF PRODUCT
00082B- 90 03 1380 BCC .2        ...DO NOT NEED TO ADD
00082D- 18 1390 CLC
00082E- 65 00 1400 ADC PLICAND
000830- 6A 1410 .2 ROR
000831- 85 04 1420 STA PRODUCT
000833- 66 02 1430 ROR PLIER    USE FOR LO-16 OF PRODUCT
000835- CA 1440 DEX
000836- D0 EE 1450 BNE .1

```


1792k 16-Bit IIc!

Why pay more for a lesser card that works in 8-Bit just because it's advertised a lot? You can buy Checkmate Technology's **State-Of-The-Art MULTIRAM IIc**™ that works great (100%) in 8-Bit, has a true Co-Processor port, & optional 16-Bit 65C816 slot saver Co-Processor card. We've lowered many prices until 1-15-86 & we'll sell you Jeeves™ at \$29, Pinpoint™ at \$49, or Supercalc 3A™ at \$119, **WITH EACH 576k OR LARGER MULTIRAM CARD. CALL FOR PRICES.**

- **MULTIRAM IIc IS A DIRECT SUBSTITUTE FOR RAMWORKS II™** or Apple Ext 80 column cards. **MULTIRAM RUNS ALL (100%) 3rd PARTY SOFTWARE** that the others do & its PAL circuit allows for changing memory mapping protocols too!
- **UP TO 768k MAIN BOARD MEMORY, UP TO 1024k PIGGYBACK BOARD MEMORY WITH FREE RGB**, sharp 80 columns, Double Hi-Res Graphics, no slot-1 interference! **A TOTAL OF 1792k MEMORY AT A MUCH LOWER PRICE!**
- **OPTIONAL 16-Bit 65C816 CO-PROCESSOR CARD** that plugs into MULTIRAM using no slots! **TRULY STATE-OF-THE-ART, BUT IT CAN'T RUN ON RAMWORKS!**
- **FREE APPLEWORKS TIME & DATE ON-SCREEN SOFTWARE** that can auto-date and auto-time stamp database files with any ProDOS clock.
- **FREE ULTRA-FAST RAM DISK SOFTWARE** that isn't like Ramworks™ \$29 Ram disk software. It can be run alone or WITH APPLEWORKS. **FREE RAM TEST & optional CP/M & Pascal Ram disk too!**
- **FREE APPLEWORKS EXPANDER SOFTWARE** that modifies AppleWorks once so all features are automatic, loads ALL or PARTS of AppleWorks into memory, runs 20 x faster, increases the Desktop over 1125k, auto-segments large files onto multiple disks, stores over 5325 records! An increased Clipboard/Word Processor update, 16-Bit integrated software AND more piggyback memory is due soon.

- **15 DAY MONEY BACK GUARANTEE, FREE SOFTWARE UPDATES, FREE 64k MEMORY WITH EACH 256k/512k CARD ONLY FROM COIT VALLEY COMPUTERS. 5 YEAR WARRANTY THAT, UNLIKE RAMWORKS, INSURES COVERAGE NO MATTER WHERE YOU BOUGHT IT! CALL FOR CURRENT PRICES, QUANTITY DISCOUNTS OR NEW FEATURES!**

**LOWER
OUR LOW-PRICE**

64k MULTIRAM IIc	155.
128k MULTIRAM IIc	173.
320k MULTIRAM IIc	187.
576k MULTIRAM IIc	237.
768k MULTIRAM IIc	287.
1024k MULTIRAM IIc/FREE RGB	388.
1280k MULTIRAM IIc/FREE RGB	499.
1536k MULTIRAM IIc/FREE RGB	548.
1792k MULTIRAM IIc/FREE RGB	598.

64k Memory Expander Chips (8)	20.
256k Memory Expander Chips (8)	50.
Pico™ Slimline Drive IIc, IIc, II+	190.
Apple IIc Enhancement Kit	62.
65C816 Co-Processor Card	162.
Slot 3 Clock	75.
Switch-A-Slot	165.

CALL FOR OTHER APPLE PERIPHERAL PRICES.

640k 16-Bit IIc!

Checkmate Technology's **State-Of-The-Art IIc** cards easily expand your IIc up to 640k, are 100% compatible with all IIc software/hardware, & come with the same **FREE SOFTWARE** as MULTIRAM IIc. **MULTIRAM C** is non-upgradable and **MULTIRAM CX** can be upgraded with a **16-Bit 65C816 Co-Processor kit!** **CALL FOR CURRENT PRICING & QUANTITY DISCOUNTS!**

- **NO JUMPER WIRES, CLIPS, OR DRIVE REMOVAL REQUIRED FOR INSTALLATION. ALL CHIPS ARE SOCKET-ED AND REMOVABLE** - unlike the competition.
- **USES ABOUT 50% LESS POWER** than the competition - causing less power supply strain or battery pack drain!
- **15 DAY MONEY BACK SATISFACTION GUARANTEE** from Coit Valley Computers. **5 YEAR WARRANTY THAT, UNLIKE THE COMPETITION, INSURES COVERAGE NO MATTER WHERE YOU BOUGHT IT!**
- **LOWER PRICES** - We sell IIc cards for much less, and our software updates are free, while competitors usually charge at least \$10.

**LOWER
OUR LOW-PRICE**

256k MULTIRAM C	229.
512k MULTIRAM C	279.
IIc Battery Pack	179.
C-VUE Flat Panel Display	449.
IIc System Clock	66.
256k MULTIRAM CX	278.
512k MULTIRAM CX	318.
16-Bit 65C816 Kit	135.

(\$10 less if bought with card)

Terms: Add \$4-Ground or \$6-Air shipping & phone # to all U.S. orders (foreign orders extra). Add 3% for P.O.'s (3% 7 net 30) & MasterCard/Visa (include #/expir). For fastest delivery send Cashier's/Certified check, Money Order, C.O.D. (add \$5) & personal checks accepted (allow 14 days). Tex res add 6 1/2% tax. **CALL FOR LATEST PRICES & QUANTITY DISCOUNTS!**

MULTIRAM/Multiview, Ramworks/Ramworks II, Pico, Jeeves, Pinpoint, Supercalc respective trademarks of Checkmate Technology, Applied Engineering, WGE Int, PBI Inc, Pinpoint Software, Sorcim

COIT VALLEY COMPUTERS

(214) 234-5047

14055 Waterfall Way

Dallas, Texas 75240

000838-	38	1460	SEC	
000839-	FB	1470	XCE	BACK TO EMULATION MODE
00083A-	60	1480	RTS	
		1490	#-----	
		1500	MULTIPLY.16X16.65802.FASTER	
00083B-	18	1510	CLC	
00083C-	FB	1520	XCE	NATIVE MODE
00083D-	C2 20	1530	REP #20	A-REG 16-BITS
00083F-	A5 00	1540	LDA PLICAND	
000841-	F0 19	1550	BEQ .3	0*ANYTHING=0
000843-	3A	1560	DEC	
000844-	85 00	1570	STA PLICAND	
000846-	A2 10	1580	LDX #16	LOOP 16 TIMES
000848-	A5 02	1590	LDA PLIER	CHECK NEXT BIT OF MULTIPLIER
00084A-	4A	1600	LSR	
00084B-	A5 04	1610	LDA PRODUCT	GET HI-16 OF PRODUCT
00084D-	90 02	1620	BCC .2	...DO NOT NEED TO ADD
00084F-	65 00	1630	ADC PLICAND	
000851-	6A	1640	ROR	
000852-	85 04	1650	STA PRODUCT	
000854-	66 02	1660	ROR PLIER	USE FOR LO-16 OF PRODUCT
000856-	CA	1670	DEX	
000857-	D0 EF	1680	BNE .1	
000859-	38	1690	SEC	
00085A-	FB	1700	XCE	BACK TO EMULATION MODE
00085B-	60	1710	RTS	
00085C-	A5 04	1720	LDA PRODUCT	INITIAL ADDEND
00085E-	85 02	1730	STA PLIER	LOW 16 OF PRODUCT
000860-	64 04	1740	STZ PRODUCT	HIGH 16 OF PRODUCT
000862-	38	1750	SEC	
000863-	FB	1760	XCE	BACK TO EMULATION MODE
000864-	60	1770	RTS	
		1780	#-----	

		1000	#SAVE S.MUL.16X16.65802.EVEN.FASTER	
		1010	#-----	
		1020	.OP 65802	
		1030	#-----	
00-		1040	A	.EQ 0.1
02-		1050	B	.EQ 2.3
04-		1060	P	.EQ 4,5,6,7
		1070	#-----	
		1080	MUL.EVEN.FASTER	
000800-	18	1090	CLC	
000801-	FB	1100	XCE	ENTER NATIVE MODE
000802-	C2 20	1110	REP #20	16-BIT A-REGISTER
000804-	64 06	1120	STZ P+2	MAKE SURE NO ADDEND IN HI-16
000806-	64 04	1130	STZ P	(DELETE IF WANT AN ADDEND IN LO-16)
000808-	A2 08	1140	LDX #8	
00080A-	80 04	1150	BRA .2	...HOP OVER SHIFTS
		1160	#-----	
00080C-	06 04	1170	.1 ASL P	DOUBLE THE PRODUCT
00080E-	26 06	1180	ROL P+2	
000810-	A5 00	1190	.2 LDA A	
000812-	29 80 00	1200	AND ##0080	LOOK AT SIGN OF LO-BYTE
000815-	F0 0B	1210	BEQ .3	...DON'T ADD MULTIPLICAND
000817-	18	1220	CLC	
000818-	A5 04	1230	LDA P	
00081A-	65 02	1240	ADC B	
00081C-	85 04	1250	STA P	
00081E-	90 02	1260	BCC .3	
000820-	E6 06	1270	INC P+2	ADD CARRY TO HI-16
		1280	#-----	
000822-	06 00	1290	.3 ASL A	SHIFT MULTIPLIER
000824-	90 0B	1300	BCC .4	
000826-	18	1310	CLC	
000827-	A5 05	1320	LDA P+1	ADD TO MIDDLE OF PRODUCT
000829-	65 02	1330	ADC B	
00082B-	85 05	1340	STA P+1	
00082D-	90 02	1350	BCC .4	
00082F-	E6 07	1360	INC P+3	(NEVER BOTHERS P+4)
		1370	#-----	
000831-	CA	1380	.4 DEX	
000832-	D0 D8	1390	BNE .1	
000834-	38	1400	SEC	
000835-	FB	1410	XCE	
000836-	60	1420	RTS	
		1430	#-----	

The MOVE routine inside the Apple //c and //e ROM transfers data conveniently to and from the auxiliary 48K area, but it does not work with the upper 16K area. Also, it does not work with the extra banks of RAM available in cards such as the RAMWORKS from Applied Engineering.

I needed that ability, so I wrote my own MOVE subroutine. Mine uses the page at \$200 in main RAM as a buffer, to simplify the movement code. If you want to move from an arbitrary bank to another arbitrary bank, my program will require you to use \$200 in main RAM as an intermediate buffer. (Somewhat like stopping at Chicago on your way from Toronto to Dallas.) My program also assumes you are always moving exactly 256 bytes (one page). This simplifies the code and the calling sequence, and is probably a reasonable restriction.

The program begins by copying itself into every bank you are using. The bank numbers must be assembled in to the list in lines 1800-1870. Notice that I use bank number \$FF to signal the main RAM, and banks from \$00 up to signal the banks of Auxiliary RAM. This code needs to reside in the same location in every bank that will be switched on, because when you move from an auxiliary bank to the main RAM that auxiliary bank will be set up so that all RAM reads come from it. This includes reads for the program, so the program had better be there.

Once the program has been initialized, you can JSR MOVE (or JSR \$C03 if you want to use a "frozen" entry point) to move a page. At the time of the JSR MOVE, you should have the high byte of the Auxiliary RAM address in the A-register, and the bank number in the X-register. Set carry (SEC opcode) to indicate moving from main \$200, or clear carry (CLC opcode) to indicate moving into main \$200. Thinking in terms of a ramdisk application, SEC for a write or CLC for a read.

Warning: my program assumes you are calling from a program that runs with the Applesoft ROM selected (see line 1780). If you plan to run it with RAM selected in the upper 16K, you will have to make appropriate changes. You could save the status of the LCRAM and LCBANK soft switches (\$C012 and \$C011 respectively) before changing them. These partially indicate the status of the \$C08x switches. You can tell whether RAM or ROM was selected, and restore the proper one after MOVE is finished. You can also tell which \$D000 bank was selected. However, you cannot tell whether the RAM was write-enabled or not; also, you cannot tell if it was in the special mode in which you read ROM and write RAM.

```

1000 *SAVE BROWN'S MOVE PROGRAM
1010 *-----
1020 * MOVE by H. Brown
1030 * Jan 18/86
1040 *-----
00-- 1050 PTR      .EQ $00.01
0200- 1060 BUFFER .EQ $200
C002- 1070 RAMRD  .EQ $C002
C004- 1080 RAMWRT .EQ $C004
C008- 1090 ALTZP  .EQ $C008
C073- 1100 BNKSEL .EQ $C073      RAMWORKS BANK SELECT REGISTER

```



DISASM 2.2e - AN INTELLIGENT DISASSEMBLER : \$30.00

Investigate the inner workings of machine language programs. DISASM converts machine code into meaningful, symbolic source. Creates a standard text file compatible with S-C, LISA, ToolKit and other assemblers. Handles data tables, displaced object code & even lets you substitute your own meaningful labels. (100 commonly used Monitor and Pg Zero names included.) An address-based triple cross reference table is provided to screen or printer. DISASM is an invaluable machine language learning aid to both novice & expert alike. Don Lancaster says DISASM is "absolutely essential" in his new ASSEMBLY COOKBOOK. For entire Apple II family including the new Apple //c (with all the new opcodes). SOURCE CODE available for an additional \$30.00

LOW LOW PRICE !!! C-PRINT For The APPLE //c : \$69.00

Connect standard parallel printers to an Apple //c. C-PRINT is a hardware accessory that plugs into the standard Apple //c printer serial port. The other end plugs into any printer having a standard 36 pin centronics-type parallel connector. Just plug in and print! High speed data transfer at 9600 Baud. No need to reconfigure serial port or load software drivers for text printing.

FONT DOWNLOADER & EDITOR : \$39.00

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Use with any Word Processor program capable of sending ESC and control codes to printer. Switch back and forth easily between standard and custom fonts. All special printer functions (like expanded, compressed etc.) apply to custom fonts. Full HIRES screen editor lets you create your own characters and special graphics symbols. Compatible with many parallel printer I/F cards. User driver option provided. For Apple II, II+, //e. Specify printer: Apple Dot Matrix, C.Itoh 8510A (Prowriter), Epson FX 80/100, or OkiData 92/93.

The Font Downloader & Editor for the Apple Imagewriter Printer. For use with Apple II, II+, //e (with SuperSerial card) and the new Apple //c (with builtin serial interface).

FONT LIBRARY DISKETTE #1 : \$19.00 Contains lots of user-contributed fonts for all printers supported by the Font Downloader & Editor. Specify printer with order.

The 'PERFORMER' CARD : \$39.00

Plugs into any slot to convert a 'dumb' centronics-type printer I/F card into a 'smart' one. Command menu eliminates need to remember complicated ESC codes. Features include perforation skip, auto page numbering with date & title. Includes large HIRES graphics & text screen dumps. Specify printer: MX-80 with Graftrax-80, MX-100, MX-80/100 with Graftraxplus, NEC 8092A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93. SOURCE CODE: \$30.00

FIRMWARE FOR APPLE-CAT: The 'MIRROR' ROM : \$25.00

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Basic modes: Dumb Terminal, Remote Console & Programmable Modem. Features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back, printer buffer, 80 col card & shift key mod support. Uses superset of Apple's Comm card and Micromodem II commands. SOURCE CODE: \$50.00

RAM/ROM DEVELOPMENT BOARD : \$30.00

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip (6116 type RAM for program development or 2716 EPROM to keep your favorite routines on-line). Maps into \$Cn00-CnFF and \$C800-CFFF.

ALL NEW !!! MIDI MUSIC PRODUCTS

MIDI means Musical Instrument Digital Interface. Use your computer with any MIDI-equipped music keyboard for entertainment and music education. Low cost MIDI player interface cable, complete with 6 song demo disk: \$49.00. Thousands of popular songs available soon on diskette (also compatible with Passport MIDI interface). Products for both the Apple IIc and Commodore 64/128. Unique general purpose MIDI expander cable and gender changer also available. Send SASE for product descriptions and prices.

Avoid a \$3.00 handling charge by enclosing full payment with order. VISA/MC and COD phone orders accepted. RAK-WARE 41 Ralph Road W. Orange NJ 07052 (201) 325-1885



```

C082-      1110 ROM      .EQ $C082
C08B-      1120 RAM1    .EQ $C08B
C083-      1130 RAM2    .EQ $C083
      1140 *-----
      1150      .OR $C00      ORG      AT BEGINING OF A PAGE
      1160 *-----
      1170 COMMONPG
OC00- 4C 06 OC 1180      JMP INIT      BRUN OR JSR TO INITIALIZE
OC03- 4C 22 OC 1190      JMP MOVE      NORMAL ENTRY
      1200 *-----
      1210 *      INIT copies COMMONPG to all 64K banks
      1220 *-----
OC06- AE 78 OC 1230 INIT      LDX BANKS      START WITH LAST 64K BANK
OC09- BD 78 OC 1240      .1      LDA BANKS,X  GET BANK #
OC0C- 8D 73 CO 1250      STA BNKSEL    SELECT 64K BANK
OC0F- 8D 05 CO 1260      STA RAMWRT+1  CHOOSE TO WRITE
OC12- A0 00      1270      LDY #0      COPY PAGE
OC14- B9 00 OC 1280      .2      LDA COMMONPG,Y
OC17- 99 00 OC 1290      STA COMMONPG,Y
OC1A- C8      1300      INY
OC1B- D0 F7      1310      BNE .2      LOOP TO END OF PAGE
OC1D- CA      1320      DEX
OC1E- D0 E9      1330      BNE .1      LOOP TO START OF TABLE
OC20- F0 46      1340      BEQ EXIT    RESTORE STANDARD MEMORY
      1350 *-----
      1360 *      enter MOVE with A = page (CX for 2nd DX)
      1370 *      X = 64K bank #
      1380 *      Carry SET for write, CLEAR for read
      1390 *-----
OC22- B0 23      1400 MOVE      BCS .3      BRANCH IF WRITING
OC24- C9 C0      1410      CMP #$C0
OC26- B0 0C      1420      BCS .1      BRANCH IF UPPER 16K
OC28- E0 FF      1430      CPX #$FF      --- READ 48K ---
OC2A- F0 15      1440      BEQ .2      SKIP IF MAIN 64K
OC2C- 8D 03 CO 1450      STA RAMRD+1  READ FROM AUX 48K
OC2F- 8E 73 CO 1460      STX BNKSEL    SELECT 64K BANK
OC32- D0 0D      1470      BNE .2
OC34- 20 98 OC 1480      .1      JSR SEL16K  --- READ 16K ---
OC37- E0 FF      1490      CPX #$FF
OC39- F0 06      1500      BEQ .2      SKIP IF MAIN 64K
OC3B- 8E 73 CO 1510      STX BNKSEL    SELECT 64K BANK
OC3E- 8D 09 CO 1520      STA ALTZP+1  SELECT AUX 16K
OC41- 18      1530      .2      CLC
OC42- 20 7E OC 1540      JSR COPYPAGE
OC45- F0 21      1550      BEQ EXIT
      1560 *-----
      1570 *      WRITING
      1580 *-----
OC47- C9 C0      1590      .3      CMP #$C0
OC49- B0 0C      1600      BCS .4      BRANCH IF UPPER 16K
OC4B- E0 FF      1610      CPX #$FF      --- WRITE 48K ---
OC4D- F0 15      1620      BEQ .5      SKIP IF MAIN 64K
OC4F- 8E 73 CO 1630      STX BNKSEL
OC52- 8D 05 CO 1640      STA RAMWRT+1  WRITING TO AUX 48K
OC55- D0 0D      1650      BNE .5
OC57- 20 98 OC 1660      .4      JSR SEL16K  --- WRITE 16K ---
OC5A- E0 FF      1670      CPX #$FF
OC5C- F0 06      1680      BEQ .5
OC5E- 8E 73 CO 1690      STX BNKSEL
OC61- 8D 09 CO 1700      STA ALTZP+1
OC64- 38      1710      .5      SEC
OC65- 20 7E OC 1720      JSR COPYPAGE
      1730 *-----
OC68- 8C 73 CO 1740 EXIT      STY BNKSEL    RESORE STD 64K FOR VIDEO
OC6B- 8D 04 CO 1750      STA RAMWRT    MAIN 48K
OC6E- 8D 02 CO 1760      STA RAMRD
OC71- 8D 08 CO 1770      STA ALTZP    MAIN 16K
OC74- AD 82 CO 1780      LDA ROM
OC77- 60      1790      RTS
      1800 *-----
      1810 *      BANKS is a table of 64K bank #'s, where
      1820 *      FF = main 64k, 00 = alt 64K when no RAMWORKS
      1830 *      00,04,08,0C = banks of a 256K RAMworks
      1840 *      1st entry is # of banks
      1850 *-----
OC78- 05      1860 BANKS      .HS 05      Five banks all told
OC79- FF 00 04
OC7C- 08 OC      1870      .HS FF.00.04.08.0C

```

```

1880 *-----
1890 * COPYPAGE copies 256 bytes
1900 * from (PTR) in specified bank to motherboard $200
1910 * or from motherboard $200 to (PTR) in specified bank
1920 *-----
1930 COPYPAGE
1940 STA PTR+1
1950 LDY #0
1960 STY PTR
1970 BCS .2
1980 .1 LDA (PTR),Y
1990 STA BUFFER,Y
2000 INY
2010 BNE .1
2020 RTS
2030 .2 LDA BUFFER,Y
2040 STA (PTR),Y
2050 INY
2060 BNE .2
2070 RTS
2080 *-----
2090 * SEL16K selects the appropriate bank in 16K area
2100 *-----
2110 SEL16K CMP #$D0
2120 BCS .1
2130 LDY RAM2 C0 -> AUX D0
2140 LDY RAM2
2150 ADC #$10
2160 RTS
2170 .1 LDY RAM1 SELECT RD/WRT RAM
2180 LDY RAM1
2190 RTS
2200 *-----

```

```

OC7E- 85 01
OC80- A0 00
OC82- 84 00
OC84- B0 09
OC86- B1 00
OC88- 99 00 02
OC8B- C8
OC8C- D0 F8
OC8E- 60
OC8F- B9 00 02
OC92- 91 00
OC94- C8
OC95- D0 F8
OC97- 60

OC98- C9 D0
OC9A- B0 09
OC9C- AC 83 C0
OC9F- AC 83 C0
OCA2- 69 10
OCA4- 60
OCA5- AC 8B C0
OCA8- AC 8B C0
OCAB- 60

```

We Make Measurement And Control Easy!

12 BIT, 16 CHANNEL, PROGRAMMABLE GAIN A/D

- All new 1984 design incorporates the latest in state-of-art I.C. technologies.
- Complete 12 bit A/D converter, with an accuracy of 0.02%!
- 16 single ended channels (single ended means that your signals are measured against the Apple's GND) or 8 differential channels. Most all the signals you will measure are single ended.
- 9 software programmable full scale ranges, any of the 16 channels can have any range at any time. Under program control, you can select any of the following ranges: ± 10 volts, ± 5 V, ± 2.5 V, ± 1.0 V, ± 500 mV, ± 250 mV, ± 100 mV, ± 50 mV, or ± 25 mV.
- Very fast conversion (25 micro seconds).
- Analog input resistance greater than 1,000,000 ohms.
- Laser-trimmed scaling resistors.
- Low power consumption through the use of CMOS devices.
- The user connector has +12 and -12 volts on it so you can power your sensors.
- Only elementary programming is required to use the A/D.
- The entire system is on one standard size plug in card that fits neatly inside the Apple.
- System includes sample programs on disk.

PRICE \$319

A few applications may include the monitoring of ● flow ● temperature ● humidity ● wind speed ● wind direction ● light intensity ● pressure ● RPM ● soil moisture and many more

A/D & D/A

A/D Features:

- Single PC card
- 8 channels A/D
- 8 channels D/A
- Superfast conversion time
- Very easy programming
- Many analog ranges
- Manual contains sample applications

A/D SPECIFICATIONS

- 0.3% accuracy
 - On-board memory
 - Fast conversion (0.78 MS per channel)
 - A/D process totally transparent to Apple (looks like memory)
 - User programmable input ranges are 0 to 10 volts, 0 to 5, -5 to +5, -2.5 to +2.5, -5 to 0, -10 to 0.
- The A/D process takes place on a continuous, channel sequencing basis. Data is automatically transferred to its proper location in the on-board RAM. No A/D converter could be easier to use.

D/A SPECIFICATIONS

- 0.3% accuracy
 - On-board memory
 - On-board output buffer amps can drive 5 MA
 - D/A process is totally transparent to the Apple (just poke the data)
 - Fast conversion (0.03 MS per channel)
 - User programmable output ranges are 0 to 5 volts and 0 to 10 volts
- The D/A section contains 8 digital to analog converters, with output buffer amplifiers and all interface logic on a single card. On-board latches are provided for each of the eight D/A converters. No D/A converter could be easier to use. The on-board amplifiers are laser-trimmed during manufacture, thereby eliminating any requirement for off-set nulling.

PRICE \$199

SIGNAL CONDITIONER

Our 8 channel signal conditioner is designed for use with both our A/D converters. This board incorporates 8 f.e.t. op-amps, which allow almost any gain or offset. For example, an input signal that varies from 2.00 to 2.15 volts or a signal that varies from 0 to 50 mV can easily be converted to 0-10V output for the A/D.

The signal conditioner's outputs are on a high quality 16 pin gold I.C. socket that matches the one on the A/D's so a simple ribbon cable connects the two. The signal conditioner can be powered by your Apple or from an external supply.

FEATURES

- 4.5" square for standard card cage and 4 mounting holes for standard mounting. The signal conditioner does not plug into the Apple; it can be located up to 1/2 mile away from the A/D.
- 22 pin 156 spacing edge card input connector (extra connectors are easily available i.e. Radio Shack).
- Large bread board area.
- Full detailed schematic included.

PRICE \$79

I/O 32

- Provides 4, 8-Bit programmable I/O Ports
- Your inputs can be anything from high speed logic to simple switches
- Any of the 4 ports can be programmed as an input or an output port
- Programming is made very easy by powerful on-board firmware
- All I/O lines are TTL (0-5 volt) compatible
- The I/O 32 is your best choice for any control application

The I/O manual includes many programs for inputs and outputs.

Some applications include:

Burglar alarm, direction sensing, use with relays to turn on lights, sound buzzers, start motors, control tape recorders and printers, use with digital joystick.

PRICE \$89

Please see our other full page ad in this magazine for information on Applied Engineering's Timemaster Clock Card and other products for the Apple.

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products are compatible with Apple II and IIe.

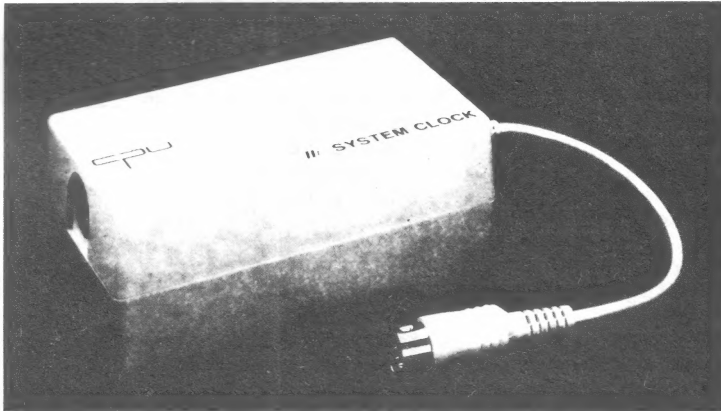
Applied Engineering's products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle three year warranty.

Texas Residents Add 5% Sales Tax
Add \$10.00 if Outside U.S.A.

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 798
Carrollton, TX 75006

Call (214) 241-6060
9 a.m. to 11 p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome
No extra charge for credit cards

**NEW
PRODUCT**



IIC SYSTEM CLOCK

- Fully ProDos compatible
- Automatic time and date stamping
- Easy to use from BASIC
- Battery operated, uses 3 "AA" batteries (will last 1-2 years before simple replacement)
- Date has year, month, date and day of week
- Time has hours, minutes and seconds
- Will time and date stamp AppleWorks files
- Will display time and date on the AppleWorks screen
- Auto access from AppleWorks data-base (just use a time and date field)
- Pass through serial port - The IIC system clock can plug into either the modem or printer serial port, then modem or printer plugs into the clock
- No hassle 5 year warranty
- Only \$79.00



"We Set the Standard"

214-241-6060

APPLIED ENGINEERING

9 AM - 11 PM

The Sep 85 (V5N12) issue of AAL included an article and program to initialize a disk with both DOS and ProDOS catalogs in separate halves of the disk. After trying to use Catalog Arranger on a disk we made with DOUBLE.INIT, we discovered that program has a bug.

The DOS catalog is written in track \$11, starting with sector 15 and going backwards to sector 1. The second and third bytes in each catalog sector are supposed to point to the next catalog sector, with the exception of those bytes in the LAST catalog sector. In the last catalog sector, the link bytes should both be \$00, to signal to anyone who tries to read the catalog that this is indeed the last sector. DOUBLE.INIT stored \$11 in the first link byte, and so some catalog reading programs such as Catalog Arranger get very confused.

The fix is to add the following lines to the program, where the line numbers correspond to those in the printed listing in AAL:

```
2201      BNE .5
2202      STY C.TRACK    (Y=0)
```

Add the label ".5" to line 2210, so that it reads:

```
2210 .5    JSR CALL.RWTS
```

If you have already created some disks with DOUBLE.INIT, we suggest you use a program such as Bag of Tricks, CIA, or some other disk zap program to clear the second byte of track \$11, sector \$01 on those disks.

An Interesting Bit of Trivia.....Bill Parker

Some time ago I asked Bob S-C if he knew the origin of the term "6502". Why was this particular number chosen? Bob didn't know, but referred me to Bill Mensch at Western Design Center.

Bill worked at Motorola and was on the design team that created the 6800, which later led to the development of the 68000. He left Motorola with a few others and formed MOS Technology (now absorbed into Commodore), where they developed a new micro-processor which was supposed to be an improved version of the 6800. Hence the decision was made to use a number in the 6000 series. As for the hundreds digit, Commodore already had chips that used just about every digit, except "5". Thus, the "6500" series of chips was born.

As history tells us, the first chip in the series, the 6501, was too close to Motorola's design, and had to be revised. The result was the 6502.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$18 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$14 postage for other countries. Back issues are available for \$1.80 each (other countries add \$1 per back issue for postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)